What is claimed is:

1. In a scheduler having at least one target processor, a method for ordering instructions in a code file having a plurality of instructions comprising:

(a) determining dependencies between instructions in said plurality of instructions;

(b) creating a directed acyclic graph showing said dependencies in said plurality of instructions, where said directed acyclic graph's nodes each correspond to an instruction from said plurality of instructions;

(c) creating at least one queue;

(d) traversing said directed acyclic graph in a dependency-preserving manner;

(e) creating a ready set of nodes by identifying which nodes in said directed acyclic graph are in a ready state and which instructions correspond to said nodes;

(f) finishing if there are no nodes found to be in a ready state;

(g) identifying said at least one queue's needs to have queue elements added in accordance with said at least one target processor;

(h) identifying said at least one queue's needs to have queue elements subtracted in accordance with said at least one target processor;

(i) choosing one of said nodes and its corresponding instruction in said ready set and one of said at least one queues such that said chosen node and said queue satisfy at least one of said queue's needs;

(j) choosing one of said nodes and its corresponding instruction in said ready set and one of said at least one queues heuristically if no node can satisfy at least one of said queue's needs;

(k) removing said chosen node from said directed acyclic graph;

(l) modifying said chosen queue in accordance with said chosen node and its associated instruction;

(m) modifying the order of instructions in said code file in accordance with said chosen node and its associated instruction; and,

(n) continuing processing at (d).

2. The scheduler of claim 1, after creating at least one queue, the method further comprising:

(a) using one queue for said at least one queue;

(b) determining a maximum desirable number of elements identifier for said one queue;

(c) choosing one of said nodes in said ready set that will add an element to said queue if said queue has fewer elements than said identifier;

(d) choosing one of said nodes in said ready set that will remove at least one element from said queue if said queue has an equal number of elements as said identifier; and,

(e) choosing one of said nodes in said ready set that will remove at least one element from said queue if said queue has more than the number of elements as said identifier;

3. The scheduler of claim 1, after creating at least one queue, the method further comprising:

(a) using one queue for said at least one queue;

(b) determining a maximum desirable number of elements identifier for said one queue; and,

(c) choosing one of said nodes in said ready set such that said queue has a total number of elements close to and not exceeding said identifier;

4. The scheduler of claim 1, after creating at least one queue, the method further comprising:

(a) using a load queue, a prefetch queue, and a store queue for said at least one queue;

(b) determining and correlating a maximum desirable number of elements identifier for each of said load queue, said prefetch queue, and said store queue;

(c) determining a precedence ordering between said load queue, said prefetch queue, and said store queue; and,

(c) choosing one of said nodes in said ready set that will change the number of elements in one of said load queue, said prefetch queue, or said store queue in accordance with said precedence order and one of said correlated identifiers.

5. The scheduler of claim 1, after creating at least one queue, the method further comprising:

(a) determining which number of queues to use in accordance with a target processor;

(a) using said determined number of queues for said at least one queue;

(b) determining a maximum desirable number of elements identifier for each of said determined number of queues;

(c) coupling a maximum desirable number of elements identifier to each of said determined number of queues;

(d) determining a precedence ordering between each of said determined number of queues;

(e) choosing one of said nodes in said ready set that will change the number of elements in one of said determined number of queues, in accordance with said precedence order and said coupled identifier, if one of said nodes can be found; and,

(f) choosing one of said nodes in said ready set that will not change the number of elements in one of said determined number of queues, in accordance with said precedence order and said coupled identifier, if none of said nodes in said ready can be found that will change the number of elements in one of said determined number of queues, in accordance with said precedence order and said coupled identifier.

6. The scheduler of claim 1, where said ordering of said instructions in said code file uses a hardware scheduler in accordance with said chosen node and its associated instruction.

7. The scheduler of claim 1, replacing modifying said chosen queue, the method further comprising:

(a) adding an element corresponding to said chosen node to said at least one queue if said at least one queue's needs is to have queue elements added; and,

(b) removing at least one element in accordance with said chosen node to said at least one queue if said at least one queue's needs is to have queue elements removed.

8. A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform a

method for ordering instructions in a code file having a plurality of instructions, the method comprising:

(a) determining dependencies between instructions in said plurality of instructions;

(b) creating a directed acyclic graph showing said dependencies in said plurality of instructions, where said directed acyclic graph's nodes each correspond to an instruction from said plurality of instructions;

(c) creating at least one queue;

(d) traversing said directed acyclic graph in a dependency-preserving manner;

(e) creating a ready set of nodes by identifying which nodes in said directed acyclic graph are in a ready state and which instructions correspond to said nodes;

(f) finishing if there are no nodes found to be in a ready state;

(g) identifying said at least one queue's needs to have queue elements added in accordance with said at least one target processor;

(h) identifying said at least one queue's needs to have queue elements subtracted in accordance with said at least one target processor;

(i) choosing one of said nodes and its corresponding instruction in said ready set and one of said at least one queues such that said chosen node and said queue satisfy at least one of said queue's needs;

(j) choosing one of said nodes and its corresponding instruction in said ready set and one of said at least one queues heuristically if no node can satisfy at least one of said queue's needs;

(k) removing said chosen node from said directed acyclic graph;

(l) modifying said chosen queue in accordance with said chosen node and its associated instruction;

(m) modifying the order of instructions in said code file in accordance with said chosen node and its associated instruction; and,

(n) continuing processing at (d).

9. The program storage device of claim 8, after creating at least one queue, the method further comprising:

(a) using one queue for said at least one queue;

(b) determining a maximum desirable number of elements identifier for said one queue;

(c) choosing one of said nodes in said ready set that will add an element to said queue if said queue has fewer elements than said identifier;

(d) choosing one of said nodes in said ready set that will remove at least one element from said queue if said queue has an equal number of elements as said identifier; and,

(e) choosing one of said nodes in said ready set that will remove at least one element from said queue if said queue has more than the number of elements as said identifier;

10. The program storage device of claim 8, after creating at least one queue, the method further comprising:

(a) using one queue for said at least one queue;

(b) determining a maximum desirable number of elements identifier for said one queue; and,

(c) choosing one of said nodes in said ready set such that said queue has a total number of elements close to and not exceeding said identifier;

11. The program storage device of claim 8, after creating at least one queue, the method further comprising:

(a) using a load queue, a prefetch queue, and a store queue for said at least one queue;

(b) determining and correlating a maximum desirable number of elements identifier for each of said load queue, said prefetch queue, and said store queue;

(c) determining a precedence ordering between said load queue, said prefetch queue, and said store queue; and,

(c) choosing one of said nodes in said ready set that will change the number of elements in one of said load queue, said prefetch queue, or said store

queue in accordance with said precedence order and one of said correlated identifiers.

12. The program storage device of claim 8, after creating at least one queue, the method further comprising:

(a) determining which number of queues to use in accordance with a target processor;

(a) using said determined number of queues for said at least one queue;

(b) determining a maximum desirable number of elements identifier for each of said determined number of queues;

(c) coupling a maximum desirable number of elements identifier to each of said determined number of queues;

(d) determining a precedence ordering between each of said determined number of queues;

(e) choosing one of said nodes in said ready set that will change the number of elements in one of said determined number of queues, in accordance with said precedence order and said coupled identifier, if one of said nodes can be found; and,

(f) choosing one of said nodes in said ready set that will not change the number of elements in one of said determined number of queues, in accordance with said precedence order and said coupled identifier, if none of said nodes in said ready can be found that will change the number of elements in one of said

determined number of queues, in accordance with said precedence order and said coupled identifier.

13. The program storage device of claim 8, where said ordering of said instructions in said code file uses a hardware scheduler in accordance with said chosen node and its associated instruction.

14. The program storage device of claim 8, replacing modifying said chosen queue, the method further comprising:

(a) adding an element corresponding to said chosen node to said at least one queue if said at least one queue's needs is to have queue elements added; and,

(b) removing at least one element in accordance with said chosen node to said at least one queue if said at least one queue's needs is to have queue elements removed.

15. A queue modeling instruction scheduler apparatus for use in compiling a program, the apparatus executable in a device having a processor operatively coupled to a memory, the apparatus comprising:

a directed acyclic graph creation module operatively disposed within said apparatus, configured to determine dependencies between instructions in a program to be compiled and to create a directed acyclic graph showing said

dependencies in said program, and where said directed acyclic graph's nodes correspond to instructions in said program;

a directed acyclic graph traversal and ready set identification module operatively disposed within said apparatus and configured to traverse said directed acyclic graph in a dependency-preserving manner and to create a ready set of nodes;

a ready set evaluation module operatively disposed within said apparatus and configured to identify which nodes in said ready set correspond to which instructions in said program, and to evaluate said instructions for their effect on memory operations;

a queue management module operatively disposed within said apparatus and configured to manage at least one queue, where managing a queue further comprises adding and removing elements from said at least one queue where said elements correspond to nodes from said directed acyclic graph;

a code scheduling module operatively disposed within said apparatus and operably connected to said program and said directed acyclic graph traversal and ready set identification module and said ready set evaluation module and said queue management module and said directed acyclic graph, configured to add and remove nodes from said directed acyclic graph and to have elements of said at least one queue added and removed and to change the order of instructions in said program in accordance with said instructions, said nodes, and said at least one queue.

16. The apparatus of claim 15 wherein said queue manager is further configured to manage a load queue, a prefetch queue, and a store queue, and wherein said code scheduling module further configured to determine and correlate a maximum desirable number of elements identifier for each of said load queue, said prefetch queue, and said store queue, and further configured to determine a precedence ordering between said load queue, said prefetch queue, and said store queue, and further configured to choose one of said nodes in said ready set that will change the number of elements in one of said load queue, said prefetch queue, or said store queue in accordance with said precedence order and one of said correlated identifiers.

17. The apparatus of claim 15 wherein said queue manager is further configured to determine what number of queues to use in accordance with a target processor and managing said determined number of queues, and wherein said code scheduling module is further configured to determine a precedence ordering between each of said determined number of queues and is further configured to choose one of said nodes in said ready set that will affect the elements in one of said determined number of queues in accordance with said precedence order.

18. The apparatus of claim 15 wherein said code scheduler module further comprises a hardware scheduler.